Buletinul Ştiinţific al Universităţii din Baia Mare
Seria B, Matematică-Informatică, vol.XIII(1997),179-188

EXTENSIONS FOR THE IMAGE BROADCASTING PROTOCOL

Ovidiu COSMA

1. Introduction

The image broadcasting protocol (IBP) has mainly didactic applications, but can be adapted for other situations that need transmissions of computer images. IBP operates in a master - slave fashion, and can improve the following activities:

– **Presentations**
The teacher's screen images can be transferred to another computer, or to all the computers handled by IBP.

– **Verifications**
The images from any computer in the network can be transferred to the teacher's screen, or to all the computers (including the teacher's).

All the operations are performed at teacher's commands.

A version of this protocol, for character modes only, is described in [1], together with two implementations. The IPX protocol [2] is used for network services.

The aim of this paper is to extend the IBP, with the possibility of transferring images in any video mode, and to reduce the amount of data transferred through the network by using image compression techniques.

2. Protocol description

If the participants to the IBP are connected through an internet, with subnets of different types, the communication functions can be implemented with a protocol situated above the network level, for example IP [3] or UDP [4]. Thus all the problems that concern getting packets from source to the destination are taken care at the inferior levels.

Because of the relatively high volume of information that has to be transferred through the network, this protocol can be more efficiently implemented for broadcast networks. In this case there is a single communication channel that is shared by all the machines on the network. Packets sent by any machine are received by all the others. An address field within the packet specifies for whom it is intended. Luckily all three IEEE 802 LAN standards fall into this category, with the exception of the IEEE 802.5 Token Ring that uses not really a broadcast medium, but a collection of individual point to point links that form a circle. More details on broadcast networks and IEEE 802 standards can be found in [5]. Thus for local area networks, the whole communication in IPB can be implemented at data link layer, because if all the computers are connected to the same LAN segment there is no need for routing. Anyhow the broadcast packets - that are destined to all possible receivers - are restricted to the segment of LAN in which the transmitter is located, and are not spawn in the whole LAN.
The remainder of this article considers the case in which all the participants to the IBP are interconnected through the same segment of LAN.

3. Message format

The messages exchanged between the master and the slaves in IBP have the following format:

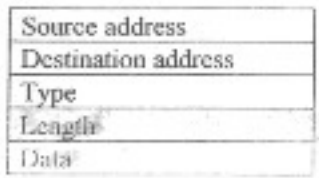

Figure 1: IBP message format

3.1. Source and destination addresses

Every participant to IBP is identified by a unique address. This address could be for example in the case of an Ethernet LAN the hardware address of the LAN interface and there is a special reserved address, with the meaning of all possible destinations (broadcast). The source address identifies the transmitter of the message and the destination address indicates the receiver(s).

3.2. Type

This is the meaning of the message and can contain the following two groups of values:

- A request for changing the operation mode of the destination. This kind of messages are generated only by the master.
- An indication concerning the content of the data field.

Here is a brief description of the four possible requests:

– **Mirror**

The receiving station(s) lock there keyboard, save the image on the screen, to be able to restore there context when ending this mode. All the images received from the network, no matter who the sender is are reproduced on the screen. Every image is preceded by a "start of frame" indication.

– **Stop**

This command determines the ending of the current activity (reproducing or sending of images). If the last mode was mirror, the context is restored and the keyboard is released.

– **Source**

The station that receives this message, sends periodically the screen images, in parallel with the current activities.

– **Ping**

This command is a request for information. The receiver(s) respond with a "ping response" indication.

The three indications are the following:

– **Ping response**

This messages contain in the data field information about the sender (the name of the operator and the mode of operation). The messages carrying data frame fragments are not very important for the IBP. Loosing such a message could pass unnoticed. On the other hand the messages that carry commands for changing the operation mode are vital. With the "ping" command the sender can verify the state of the receiver(s) before going on with the protocol. This solution is an alternative to the traditional acknowledgements that imply among other things counting and labelling the messages.

– **Frame fragment**

The data field of this messages contains a frame fragment that is immediately decoded and displayed on the screen if the receiver is in the "mirror" mode. (If pixel reordering is used, the whole image must be received before the decoding process can begin.) The length of packages is limited in every network by the Maximum Transfer Unit (MTU) for example the MTU for Ethernet is 1500 bytes. Thus the frames must be fragmented to be able to travel through the network, and the fragments must be labelled for the frame rebuild process. If the IBP works in an internet and the implementation uses IPX or UDP, there is no guarantee for delivery, order and accuracy of the packets. Thus in this case IBP has to solve this problems also. The use of a reliable transport level protocol is not possible because of the large amount of data that must be transferred.

Within any segment of LAN, packet reordering is not possible, and communication errors are rarely encountered, so the implementation of IBP can be simplified without visible deficiencies.

– **Start of frame**

The receiving station(s) that are in the mirror mode, update their video interface operation mode, according to the information contained in the data field.

3.3. Length

Indicates the length of the data field.

3.4. Data

The data field can have three different formats, corresponding to the three possible indications. The contents of the data field for this message types is shown below. The request messages have no data field.

– **Ping response**

Operator name
Current operation mode

– **Frame fragment**

No. of frame
Fragment offset
Fragment data

Figure 2: The data field for different indications

– **Start of frame**

No. of frame
Video mode
Screen resolution
No. of colours
Motion compensation
Prediction
Pixel ordering
Run length encoding
Huffman coding

This message is inserted by the transmitter at the beginning of every frame. If the receiver of this message is in the "mirror" mode, it prepares for receiving a new frame. The video mode indicates if the next frame will be in character mode or in graphic mode. Thus it is possible for the IBP to reproduce exactly the images of the transmitter, even if it switches between different video modes.

The screen resolution is measured in pixels for the graphic modes, and in characters for the alphanumeric ones.

The number of colours determine the number of bits / pixel. The receiver switches - if necessary - the video mode according to the information contained in this message.

The other fields indicate the compression techniques used for the next frame. This techniques will be described in the following paragraphs.

4. Compression

The volume of information transferred by IBP can be substantially reduced by compression techniques. Digital images can be compressed by eliminating redundant information. There are three types of redundancy that can be exploited by image compression techniques:

– **Spatial Redundancy**

In almost all natural images, the values of neighbouring pixels are strongly correlated. The correlation is even stronger in the case of computer images, that contain mainly text and no noise.

– **Spectral redundancy**

In images composed of more than one spectral band, the spectral values for the same pixel location are often correlated.

– **Temporal redundancy**

Adjacent frames in a sequence often show very little change.

The removal of spatial and spectral redundancies can be accomplished by reversible transformations that decorrelate the data. Temporal redundancy is exploited by techniques that only encode differences between adjacent frames, such as motion prediction and compensation. The frames built from the previous ones, will be called intercoded (P) and a frames coded without motion compensation, that is, using only transform coding will be called intracoded (I).

P frames can substantially improve compression, because there is usually little difference between computer image frames, but they have the disadvantage that the errors from one frame are propagating to all the others. IBP inserts an I frame at every change of video mode and after every n P frames, to eliminate such errors. The percentage of P frames can be higher in local area networks, because of the low error rate.

4.1. Image coding in character modes

Though a typical 80 x 25 character image contains only 4000 bytes, and in LANs IPB can operate without compression because of the high bit rate, in the case of long distance connections through noisy telephone lines, compression is required.

IPB uses at most three techniques to compress character images:

4.1.1. Separation

The computer image content is split in two components: a character image and a display

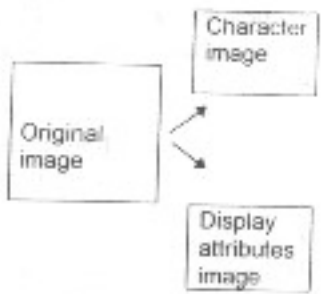

Figure 3:
Character image separation

attributes image. This increases the efficiency of the run length encoding, even without motion compensation, because the two images usually contain runs of repeated symbols.

4.1.2. Motion compensation

This technique exploits the fact that a frame F_i is likely to be similar to its predecessor F_{i-1}, and so can be nearly constructed from it.

The images are divided into 5 x 10 character blocks, and for each block is searched the best possible match in a region surrounding the block in the previous frame. The amount of x and y translation for the best match is called the motion vector. If the frame rate is high, it is sufficient to restrict the motion vector search to the range [-2, 2]. This technique will require additional buffering and introduces delay in both encoding and decoding. The criteria for "best matching" is determined by the encoder. Then an error image is constructed by subtracting the original blocks from the ones indicated by the motion vectors. If the two blocks don't match closely, the motion vectors are replaced by an invalid value (outside the accepted range), and the computing of the error block is disabled, because otherwise the subtraction could generate rare values, outside the alphanumeric character set, that would compromise the next operations.

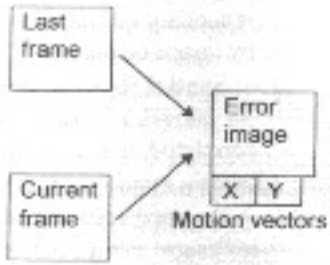

Figure 4: Motion compensation

If at this step, the new frame is found to be identical with the previous one (all motion vectors are null), the whole compression process stops, and no frame is transmitted to the receiver(s).

The motion vectors are computed only for the character image, but most of them should be valid for the display attributes image also. An error image for display attributes is build according to this motion vectors.

Motion compensation can produce a big improvement in computer image compression, because scrolling and dragging are frequent operations, and usually exist a lot of similarities between consecutive frames.

The previous steps map the image into a transform domain which is more easily encoded by the following methods. Until now no compression was performed. In fact the length of the image increased, because of the motion vectors.

4.1.3. Run length encoding

This technique replaces the runs of repeated symbols with a special escape code, a count and the symbol. Because of the character codes subtractions in the motion compensation step, the display attributes image, and the character image can contain any 8 bit combinations. Therefore if the escape code appears anywhere in

the text, it must be doubled to prevent confusion. The procedure is called character stuffing [5]. In order to increase the probability of encountering long runs of zeros, the characters are read from the image buffers in block order. This reordering takes full advantage of the previous step.

4.1.4. Huffman coding

This last transformation performs redundancy reduction using variable length code words, and will be described later.

4.2. Image coding in graphic modes

In most situations computer images contain text printed in different ways on the screen. The total absence of noise makes them very suitable for redundancy reduction techniques.

There are many redundancy reduction schemes, but most of them were developed for natural images, where a certain amount of distortions is usually not observed by the human eye. The most important such lossy techniques are based on discrete cosine transform (DCT), vector quantization and wavelet theory [6,7,8]. In the case of computer images, the quality can be seriously altered by such approximations. On the other hand the efficiency of DCT is low because of the high contrast and sharp edges.

The IBP needs a lossless compression algorithm, that is simple enough to run in real time and has a high compression ratio. At first sight, it seems almost impossible to achieve such antagonistic specifications, but in the case of LANs, the compression ratio can be very much improved by increasing the percentage of P frames. This is possible due to the low error ratio. On the other hand, because of the static nature of computer images the P frames can be very efficiently encoded.

The compression principles in character and graphic modes are the same, thus the following paragraphs will focus only on the differences.

In the case of graphic video modes, all three colour planes (RGB) are processed separately. Each colour plane contains a grey scale image. The number of bits per pixel results from the maximum number of colours in the video mode included in the start of frame indication. The compression of images involves some of the following techniques:

4.2.1. Motion compensation

The block dimension is 16 x 16, and the motion vector range is [-14, 14]. The criteria for finding the motion vectors is minimising the average pixel value in the error image. If the current frame is found to be identical with the previous one, the whole compression process is stooped and nothing is transmitted.

4.2.2. Prediction

The idea is to form an error image by comparing the value of each original pixel to the value given by a prediction function. This technique is useful for I frames that are not motion compensated. The prediction efficiency can be very high for computer images, where solid areas filled with the same colour are frequent. The prediction is based on the values of certain (fixed) neighbouring pixels. These pixels have already been encoded and are therefore known to the decoder. The prediction is thus identical in the encoding and decoding phases. The number of the neighbouring pixels gives the degree of the prediction function.

The lossless JPEG predictor produces results which, in light of its simplicity, are surprisingly close to the state of the art for lossless continuous tone compression. The JPEG predictor uses the following 3 pixel prediction context:

A	B
C	X

A, B and C are the pixels within the context
X is the pixel to be coded

Figure 5: Prediction context

A higher context brings unnecessary complexity without remarkable improvements. A predictor combines the values of up to three neighbouring pixels (A, B and C) to form a prediction of the pixel indicated by X in figure 5. This prediction is then subtracted from the actual value of sample X. Any of the predictors in figure 6 can be used.

0	no prediction
1	A
2	B
3	C
4	A+B-C
5	A+((B-C)/2)
6	B+((A-C)/2)
7	(A+B)/2

Figure 6: Predictors

The encoder can use any source image precision from 2 to 16 bits / sample. The description of JPEG compression standard can be found in [9].

The IBP uses an original adaptive prediction algorithm, based on the JPEG model.

For each context, the mean value ($\overline{m}$) and absolute central moment ($\overline{\alpha}$) approximations are calculated with the following formula:

$$\overline{X} = \frac{1}{3}\left(\overline{A} + \overline{B} + \overline{C}\right), \quad \overline{\alpha} = \frac{1}{3}\left(\left|\overline{A} - \overline{X}\right| + \left|\overline{B} - \overline{X}\right| + \left|\overline{C} - \overline{X}\right|\right)$$

where $\overline{A}$, $\overline{B}$ and $\overline{C}$ are the most significant 4 bits of the three pixel values.

The algorithm maintains for each possible combination of $\overline{m}$ and $\overline{\alpha}$, and for each predictor f_i, an estimate of the probability p_i that the predictor f_i gives the best result. For each context, the pair m and α is calculated, the predictor with the highest associated probability is used, and then the probabilities p_i are updated. This algorithm has to make a compromise between the need to quickly adapt the estimates p_i to image statistics, and the need to maintain good estimates. This problem can be solved by considering only the last n predictions for keeping up

the estimates. The best value for n can be determined experimentally for certain types of images.
The disadvantage of adaptive prediction is the increased running time.
Other prediction techniques can be found in [11]

4.2.3. Pixel reordering

The predictive encoding can be combined with pixel reordering. The idea is to rearrange the pixels of each row, so that the null error pixels are separated from the other ones. Each prediction context is classified either GOOD or BAD according to the frequency of exact predictions. Thus there can be fixed a treshold frequency (for example 90%) in the prediction model, and refer all contexts with higher frequencies of exact predictions as GOOD and the remaining ones as BAD. The pixels of the error image are reordered so that all the all the pixels resulting from GOOD predictions are stacked by starting from the left border of the row, and those in the BAD category from the right border towards the beginning of the row. After this reordering is expected that the null error pixels are prevailing in the beginning of the rows, thus increasing the efficiency of the run length encoding. Although the pixels are mixed in the reordering phase, it is still possible to decode each row correctly. This is because the coding context is known while decoding a particular pixel and thus the decoder knows if the next pixel is to be picked from the left or the right stack.
A variant of this technique for binary images is presented in [10], with the difference that block coding is used instead of run length encoding in the next step.

4.2.4. Run length encoding takes advantage of the runs of consecutive zeros created by motion compensation or prediction. If pixel reordering is activated, (in I frames) the pixels in the error image are read in column major order, to increase the chance of finding long runs of zeros. In this case there is a disadvantage because the receiver has get the whole image data before decoding can start. If motion compensation is used (in P frames) the pixels are read in block order.

4.2.5. Huffman coding

The Huffman codex [5,12] is used to compress the data closer to symbol entropy. One reason for using the Huffman coder is that it is relatively easy to implement. To compress data symbols, the Huffman coder creates shorter codes for frequently occurring symbols and longer codes for occasionally ones. However the theoretical compression limit is impossible to achieve with independently coded symbols, due to the need to represent each symbol in an integral number of bits.
Higher compression rates can be achieved by using the arithmetic coder, that drops the requirement that each symbol be individually coded. The arithmetic

coder has the disadvantage that it consumes significantly more computing power and memory than the Huffman coder. Descriptions of the arithmetic coder can be found in [5,12].

REFERENCES

1 *O. Cosma,* An image broadcasting protocol, Bul. Şt. Univ. Baia Mare, vol XII, No 2, 1996, 257 - 266.

2 *V. Cristea, N Ţăpuş, T. Moisa, V. Damian,* Reţele de calculatoare, Teora, 1992.

3 *IETF,* RFC 1883: Internet Protocol, Version 6 (Ipv6) Specification.

4 *IETF,* RFC 768: User Datagram Protocol.

5 *Andrew S. Tanenbaum,* Computer Networks, Prentice Hall International, Inc. 1989.

6 *International Telecommunication Union (ITU),* Recommendation H.263,Video Coding for Low Bitrate Communication, Mai 1996.

7 *P.C. Cosman, R.M. Gray,* Using Vector Quantization for Image Processing.

8 *M.L. Hilton, B.D. Jawerth,* Compressing Still and Moving Images with Wavelets, Multimedia Systems vol 2, no 3, 1994.

9 *G.K. Wallace,* The JPEG Still Picture Compression Standard, IEEE Transactions on consumer Electronics 12-1991.

10 *P.Franti, O. Nevalainen,* Compression of Binary Images by Composite Methods Based on the Block Coding, Univ. of Turku, Finland.

11 *CCITT, ISO/IEC,* CCITT Draft Recommendation T.82, ISO/IEC Draft International Standard 11544, Coded Representation of Picture and Audio Information - Progressive Bi-level Image Compression, April 1992.

12 *Ty Newton,* Image Compression Literature Review, April 1995.

Received 15.06.1997

North University of Baia Mare
Department of Mathematics and Computer Science
Victoriei 76, 4800 Baia Mare
ROMANIA
E-mail: ovidiu@univer.ubm.ro